Complexity of Acyclic Term Graph Rewriting*

Martin Avanzini[1] and Georg Moser[2]

1 **Università di Bologna, Italy & INRIA Sophia Antipolis, France**
martin.avanzini@uibk.ac.at
2 **Universität Innsbruck, Austria**
georg.moser@uibk.ac.at

Abstract

Term rewriting has been used as a formal model to reason about the complexity of logic, functional, and imperative programs. In contrast to term rewriting, *term graph rewriting* permits sharing of common sub-expressions, and consequently is able to capture more closely reasonable implementations of rule based languages. However, the automated complexity analysis of term graph rewriting has received little to no attention.

With this work, we provide first steps towards overcoming this situation. We present adaptions of two prominent complexity techniques from term rewriting, viz, the *interpretation method* and *dependency tuples*. Our adaptions are non-trivial, in the sense that they can observe not only term but also graph structures, i.e. take sharing into account. In turn, the developed methods allow us to more precisely estimate the runtime complexity of programs where sharing of sub-expressions is essential.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, F.3.2 Semantics of Programming Languages, F.4.1 Process Management, F.4.2 Storage Management

Keywords and phrases Program Analysis, Graph Rewriting, Complexity Analysis

Digital Object Identifier 10.4230/LIPIcs.FSCD.2016.10

1 Introduction

In recent years automated complexity analysis of term rewriting (see [20] for an early overview) has received increased attention, which has manifested itself in a number of significant breakthroughs. For brevity, we only mention recent work on direct methods [21, 3, 29], on worst case lower bounds [15], and on certification [6]. Furthermore the liveliness of the designated complexity competition clearly showcases the various activities in this area. [1] These activities have also triggered applications outside of rewriting. In particular term rewriting has been very successfully used as a formal model to reason about the complexity of logic, functional, and imperative programs, cf. [16, 28, 11, 2].

In contrast to term rewriting, *term graph rewriting* permits sharing of common sub-expressions, and consequently is able to capture more closely reasonable implementations of rule based languages. However, the automated complexity analysis of term graph rewriting has received little to no attention. This is somewhat surprising. On the one hand, term graph rewriting is typically motivated as *implementation* of term rewriting. Hence effectivity of the implementation should have been an issue. On the other hand, (term) graph rewriting is the rule in any kind of implementation of functional programs [27]. Consider for instance the

* This work was partially supported by FWF (Austrian Science Fund) projects J 3563, P 25781-N15, and by AFRL/DARPA.

[1] http://www.termination-portal.org/wiki/Termination_Competition.

```
power x 0 = 1
power x n = y * y * (if n `mod` 2 == 0 then 1 else x)
  where y = power x (n `div` 2)
```

Figure 1 Fast Exponentiation in `Haskell`.

`Haskell` program depicted in Figure 1. The example showcases the necessity to implement non-strict evaluation via graph reduction. Indeed, if we assign unit cost to arithmetic operations as usual, then it is easy to see that graph reduction in general requires time linear in the bit-length of n. In contrast, when implemented naively, e.g. as a *term rewrite system* (*TRS* for short) under an outermost reduction strategy, in each step the recursive call is duplicated. Conclusively the runtime complexity becomes exponential in the setting of TRSs. Moreover, transformations from imperative languages to term rewriting would profit from a more direct representation of the heap as a graph, rather than a tree. Thus complexity analysis of term graph rewriting, automated if possible, should have significant impact.

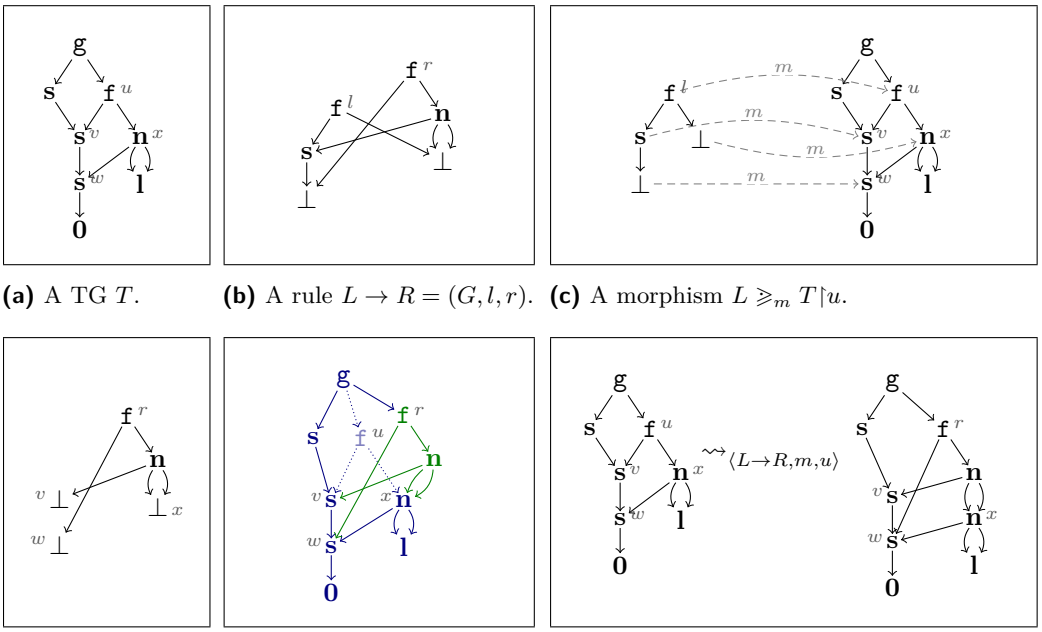
In this paper, we provide first steps towards overcoming this situation. We present adaptions of two prominent complexity techniques from term rewriting, viz, the *interpretation method* and *dependency tuples*. We summarise the contributions of this paper.

- We clarify and fix the notion of *runtime complexity* in the context of term graph rewriting (see Section 3). This is a non-trivial task, as we have to take care of succinct representations of start terms.
- We provide a novel *interpretation method* for term graph rewrite systems (Theorem 12). This method is obtained by a careful adaption of the notion of well-founded monotone algebra to term graphs.
- We show that in the context of sharing, existing restrictions of the dependency tuple approach to innermost evaluation can be overcome and establish a *dependency pair method* for term graph rewrite systems (Theorem 21).

The results above transfer two core techniques of the dependency pair framework to the complexity analysis of term graph rewrite systems. Great care has been taken to establish the correctness of these techniques in the more general setting of relative graph rewriting. Consequently, these methods are readily applicable in the *complexity pair framework* from [4], suited to term graph rewrite systems. Although not presented here, this in turn paves the way to transfer with relative ease a variety of complexity techniques applicable in the dependency pair setting from term to graph rewriting, notably, the *usable rules criterion*, *predecessor estimation*, *dependency graph decomposition* and various *simplification techniques*, see [4].

Our adaptions are non-trivial, in the sense that they can observe not only term but also graph structures, i.e. take sharing into account. In turn, the developed methods allow us to more precisely estimate the runtime complexity of programs where sharing of sub-expressions is essential.

This paper is structured as follows. In the next two sections we cover basics and clarify the notion of term graph rewriting employed. In Section 4, we present our interpretation method of term graph rewriting and in Section 5, we adapt the dependency tuple technique to this context. In Section 6 we discuss related work. Finally, in Section 7 we conclude and mention future work.



(a) A TG T. **(b)** A rule $L \to R = (G, l, r)$. **(c)** A morphism $L \geqslant_m T\restriction u$.

(d) The TG $m(R)$. **(e)** The TG $T\langle m(R)\rangle_u$. **(f)** A reduction step on T with rule $L \to R$.

Figure 2 Step-by-step construction of a graph rewrite step.

2 Preliminaries

We shortly recap basic definitions and notions. With $\mathbb{N}$ we denote the set of natural numbers $\{0, 1, 2, \dots\}$. For a set S, we denote by S^* the set of finite sequences $[s_1, s_2, \dots, s_n]$ over elements $s_i \in S$. A partial function f from A to B is denoted by $f : A \nrightarrow B$. Its *domain* is $\mathsf{dom}(f) := \{a \in A \mid f(a) \text{ is defined}\}$. For two partial functions $f, g : A \nrightarrow \mathbb{N}$ and $a \in A$ we define $f(a) \leqslant_{\mathsf{k}} g(a)$ if either $f(a)$ or $g(a)$ is undefined, or $f(a)$ and $g(a)$ are defined and $f(a) \leqslant g(a)$ holds.

Let $\to \subseteq S \times S$ be a binary relation. We denote by $\to^+$ the transitive and by $\to^*$ the transitive and reflexive closure of $\to$. We say that $\to$ is *well-founded* or *terminating*, if there is no infinite sequence $s_0 \to s_1 \to \dots$. It is *finitely branching*, if the set $\{t \mid s \to t\}$ is finite for each $s \in S$. For two binary relations $\to_A$ and $\to_B$, the relation of $\to_A$ *relative* to $\to_B$ is defined by $\to_A/\to_B := \to_B^* \cdot \to_A \cdot \to_B^*$. The *derivation height* $dh_\to : S \nrightarrow \mathbb{N}$ with respect to $\to$ over S is defined by $dh_\to(s) := \max\{\ell \in \mathbb{N} \mid s = s_0 \to s_1 \to \cdots \to s_\ell\}$. Note that $dh_\to$ is total whenever $\to$ is terminating and finitely branching. Let $(S_i)_{i\in\mathbb{N}}$ denote a countably infinite family of monotonically increasing subsets of S, i.e. $S_i \subseteq S_{i+1}$ for all $i \in \mathbb{N}$, whose limit is S. For brevity, we denote the family $(S_i)_{i\in\mathbb{N}}$ by S. We define $\mathrm{rc}^S_\to : \mathbb{N} \nrightarrow \mathbb{N}$ by $\mathrm{rc}^S_\to(n) := \max\{dh_\to(s) \mid s \in S_n\}$.

3 Term Graph Rewriting and All That

We introduce central concepts and notions of term graph rewriting, see [7] for an overview.

Term Graphs

Let $\mathcal{F}$ denote a signature, i.e. a finite set of function symbols. Each $\mathtt{f} \in \mathcal{F}$ is associated with a natural number $\mathsf{ar}(\mathtt{f})$, its *arity*. Moreover, we suppose a partitioning of the signature $\mathcal{F}$

into *defined symbols* $\mathcal{D}$ and *constructors* $\mathcal{C}$. Defined symbols take over the role of *operations*, whereas constructors are used to build *values*. Throughout the following, the signature $\mathcal{F}$ and its separation into $\mathcal{D}$ and $\mathcal{C}$ is kept fixed. A *term graph* (*TG* for short) T over the signature $\mathcal{F}$ is a *directed acyclic* graph whose internal nodes are labeled by symbols in $\mathcal{F}$, and where outgoing edges are ordered. Formally, T is a triple $(N_T, \mathsf{suc}_T, \mathsf{lab}_T)$ consisting of *nodes* N_T, a *partial successors function* $\mathsf{suc}_T : N_T \nrightarrow N_T^*$ from nodes to sequences of nodes, and a *partial labeling function* $\mathsf{lab}_T : N_T \nrightarrow \mathcal{F}$. Unlabeled nodes take the role of variables in terms, and are collected in $V_T \subseteq N_T$. We require that TGs are *compatible with their labeling*, in the sense that for each node $u \in N_T$, if $\mathsf{lab}_T(u) = \mathtt{f}$ then $\mathsf{suc}_T(u) = [u_1, \dots, u_{\mathsf{ar}(\mathtt{f})}]$ and otherwise, $\mathsf{suc}_T(u)$ is undefined. In the former case, we also write $T(u) = \mathtt{f}(u_1, \dots, u_{\mathsf{ar}(\mathtt{f})})$. We define the *successor relation* $\rightharpoonup_T$ on nodes in T such that $u \rightharpoonup_T v$ holds if $\mathsf{suc}_T(u)$ is defined and v occurs in $\mathsf{suc}_T(u)$. If v occurs at the i^{th} position we also write $u \xrightarrow{i}_T v$. Throughout the following, we consider only *acyclic* TGs, that is, we demand that $\rightharpoonup_T$ is acyclic. If not mentioned otherwise, we also suppose that TGs are *rooted*, i.e. T contains a unique node $\mathsf{rt}(T) \in N_T$, the *root*, from which all nodes $v \in N_T$ are *reachable*: $\mathsf{rt}(T) \rightharpoonup_T^* v$. See Figure 2(a) for an example of a TG. Here, we depict nodes directly by their label, possibly annotating node identities. *Unfolding* a term graph T from its root results in a finite term over the signature $\mathcal{F}$ and variables V_T, and we sometimes use this term as a linear representation for the TG T. For instance, the TG depicted in Figure 2(a) unfolds to $\mathtt{g}\big(\mathbf{s}(\mathbf{s}(\mathbf{s}(\mathbf{0}))), \mathtt{f}(\mathbf{s}(\mathbf{s}(\mathbf{0})), \mathbf{n}(\mathbf{s}(\mathbf{0}), \mathbf{l}, \mathbf{l}))\big)$.

The *size* $|T|$ of the TG T refers to the cardinality of N_T. The TG T is called *ground* if $V_T = \varnothing$. For a subset $\mathcal{G} \subseteq \mathcal{F}$ of function symbols, we collect in $N_T^{\mathcal{G}} \subseteq N_T$ all nodes u with $\mathsf{lab}_T(u) \in \mathcal{G}$. We call a node $u \in N_T$ *below* and *above*, respectively, of a node $v \in N_T$ in accordance to the topological ordering induced by $\rightharpoonup_T^*$. Two nodes are called *parallel* in T, if they are mutually unreachable. For instance, in Figure 2(a), the nodes v and x are parallel. The TG T is a *tree* if every node in T is reachable by precisely one path from its root $\mathsf{rt}(T)$. Thus, trees do not exhibit sharing.

We denote by $T\restriction u$ the *sub-graph* of T *rooted* at node $u \in N_T$. With $T[v \leftarrow u]$ we denote the graph obtained by *redirecting* all edges pointing to u to point to the node v. That is, $T[v \leftarrow u]$ denotes the TG with nodes $N_T \cup \{v\}$, labeling $\mathsf{lab}_{T[v\leftarrow u]} := \mathsf{lab}_T$ and the successor function $\mathsf{suc}_{T[v\leftarrow u]}$ is defined such that (i) $w \xrightarrow{i}_{T[v\leftarrow u]} v$ holds for each edge $w \xrightarrow{i}_T u$ in T, and (ii) $w \xrightarrow{i}_{T[v\leftarrow u]} w'$ holds whenever $w \xrightarrow{i}_T w'$ for $w' \neq u$. Note that if $v \notin N_T$, then v is considered a variable node in $T[v \leftarrow u]$. The notion is naturally extended to sequences, i.e. $T[v_1, \dots, v_n \leftarrow u_1, \dots, u_n]$ denotes the TG obtained by redirecting edges pointing to u_i to v_i for all $1 \leqslant i \leqslant n$. Here, we assume that for all $1 \leqslant i, j \leqslant n$, the node u_i is distinct from u_j (if $j \neq i$) and v_j. We denote by $S \cup T$ the *union* of two TGs S and T. To avoid ambiguities, we require that if $u \in N_S \cap N_T$ then $\mathsf{lab}_S(u)$ or $\mathsf{lab}_T(u)$ is undefined. We define

$$\mathsf{suc}_{S\cup T}(u) := \begin{cases} \mathsf{suc}_S(u) & \text{if } u \in N_S \text{ and } \mathsf{lab}_S(u) \in \mathcal{F}, \\ \mathsf{suc}_T(u) & \text{if } u \in N_T \text{ and } \mathsf{lab}_T(u) \in \mathcal{F}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Similarly, we define the labeling $\mathsf{lab}_{S\cup T}$. We write $T\langle S\rangle_u$ to denote the replacement of the subgraph in T at node u by S:

$$T\langle S\rangle_u := \begin{cases} S & \text{if } u = \mathsf{rt}(T), \\ \big(T[\mathsf{rt}(S) \leftarrow u] \cup S\big)\restriction \mathsf{rt}(T) & \text{otherwise.} \end{cases}.$$

For two rooted TGs $S = (N_S, \mathsf{suc}_S, \mathsf{lab}_S)$ and $T = (N_T, \mathsf{suc}_T, \mathsf{lab}_T)$, a mapping $m : N_S \to N_T$ is called *morphic* in $u \in N_S$ if (i) $\mathsf{lab}_S(u) = \mathsf{lab}_T(m(u))$ whenever $\mathsf{lab}_S(u)$ is defined, and

(ii) $u \xrightarrow{i}_S v$ implies $m(u) \xrightarrow{i}_T m(v)$ for all appropriate i. A (*rooted*) *homomorphism* from S to T is a mapping $m : N_S \to N_T$ that (i) maps the root of S to the root of T and that (ii) is morphic in all nodes $u \in N_S^{\mathcal{F}}$. We write $S \geqslant_m T$ to indicate that m is a homomorphism from S to T. We denote by $\underline{m}$ its extension outside of its domain, such that $\underline{m}(u) = u$ whenever u is not in the domain of m. Two TGs are *isomorphic*, in notation $S \cong T$, if there exist two morphisms m_1, m_2 with $S \geqslant_{m_1} T$ and $T \geqslant_{m_2} S$, or equivalently, if $S \geqslant_m T$ holds for a bijective morphism m. Observe that two isomorphic TGs are equal up to renaming of nodes.

Term Graph Rewriting

A *graph rewrite rule* over the signature $\mathcal{F}$ is a triple (G, l, r) where G is a TG over $\mathcal{F}$ and $l, r \in N_G$ are two distinguished nodes, denoting the root of the left- and right-hand side, respectively. We require that all nodes in G are reachable from l or r. This way, a graph rewrite rule can be denoted by $L \to R$, for the *left-hand side* $L := G\restriction l$ and *right-hand side* $R := G\restriction r$. Furthermore, we demand that the root l of the left-hand side L is labeled by a defined symbol, and that all variable nodes in R occur also in L. The label of l is called the *defined symbol* of $L \to R$. The maximal sub-graph of G contained in both L and R is called the *interface* of the rule $L \to R$. See Figure 2(b) that depicts a graph rewrite rule defining $\mathtt{f}$, with left-hand side $\mathtt{f}(\mathbf{s}(x), y)$ and right-hand side $\mathtt{f}(x, \mathbf{n}(\mathbf{s}(x), y, y))$, for x and y denoting the two variable nodes. The interface of this rule consists of the two variable nodes, as well as the node labeled by $\mathbf{s}$. Isomorphisms are naturally extended to graph rewrite rules. An isomorphic copy of a rule $L \to R$ is also called a *renaming*. If R lies within L, we also say that $L \to R$ is a *collapsing* rule. A *graph rewrite system* (*GRS* for short) $\mathcal{G}$ over $\mathcal{F}$ is a finite set of graph rewrite rules over $\mathcal{F}$.

Let S be a ground TG and let $L \to R$ be a graph rewrite rule with nodes disjoint from those in S. Application of $L \to R$ on S involves the identification of a *redex*, i.e. homomorphic copy of L in S, replacing this copy with a copy of R, retaining interface nodes, and finally *garbage collecting* nodes that became inaccessible. All of this is formalised as follows. We say that the graph rewrite rule $L \to R$ *matches* the TG S at node u if $L \geqslant_m S\restriction u$ holds. The triple $\langle L \to R, m, u\rangle$ is called a *redex* in S, and the node u of S the *redex node*, compare Figure 2(c). With $m(R)$ we denote the *instantiation* of the right-hand side of R by the matching morphism $L \geqslant_m S\restriction u$. This is done by redirecting edges according to the morphism m, from R to S, and then removing inaccessible nodes. Formally, let $u_1, \dots, u_n$ denote all nodes of the interface in $L \to R = (G, l, r)$. Then

$$m(R) := R[m(u_1), \dots, m(u_n) \leftarrow u_1, \dots, u_n]\restriction \underline{m}(r) \,.$$

Observe that $\underline{m}(r) \neq r$ only when R is collapsing, i.e. when r is an interface node u_i. In this case, $m(R)$ consists of the single variable node $m(r)$. Compare Figure 2(d) which depicts $m(R)$ with respect to the right-hand side R of the rule from Figure 2(b) and the matching morphism drawn in Figure 2(c).

We define $S \rightsquigarrow_{\langle L\to R,m,u\rangle} T$, if $\langle L \to R, m, u\rangle$ is a redex in S and $T := S\langle m(R)\rangle_u$, and call $S \rightsquigarrow_{\langle L\to R,m,u\rangle} T$ a *pre-reduction step*. We write $S \rightsquigarrow_{L\to R} T$ when the precise redex is unimportant. Since nodes of S and R are disjoint, $S\langle m(R)\rangle_u$ is well-defined and acyclic. Observe that by construction, the right-hand side R is embedded in T at node $\mathsf{rt}(R)$, more precise, $R \geqslant_{\underline{m}} T\restriction \mathsf{rt}(R)$ holds. Compare Figures 2(e) and 2(f).

For a rule $L \to R = (G, l, r)$, we define their *difference set* $\Delta(L \to R) := (N_R \setminus N_L) \cup \{l\}$ if $l \in N_R$, and $\Delta(L \to R) := N_R \setminus N_L$ otherwise. For symbols $\mathcal{D} \subseteq \mathcal{F}$ we denote by $\Delta^{\mathcal{D}}(L \to R)$ the restriction of $\Delta(L \to R)$ to nodes labeled by symbols from $\mathcal{D}$. The following technical result will be useful later.

▶ **Proposition 1.** *If* $S \rightsquigarrow_{\langle L\to R,m,u\rangle} T$ *then* $N_T^{\mathcal{D}} \subseteq (N_S^{\mathcal{D}} \setminus \{u\}) \cup \{\underline{m}(v) \mid v \in \Delta^{\mathcal{D}}(L \to R)\}$.

Proof. Observe that $\underline{m}$ is the identity function on $N_R \setminus N_L$, whereas $\underline{m}(l) = u$ for l the root of the left-hand side L. Further, $N_T^{\mathcal{D}} \subseteq N_S^{\mathcal{D}} \cup \Delta^{\mathcal{D}}(L \to R) \setminus \{l\}$ where moreover, $u \notin N_T^{\mathcal{D}}$ whenever l does not occur in the right-hand side R. From this, the claim follows by case analysis on $l \in N_R$. ◀

To every TG T, we can identify an isomorphic, *canonical* TG $\mathcal{C}(T)$ where nodes are sets of *positions*, i.e. finite sequences of integers [26]. In particular, if two TG S and T are isomorphic, then $\mathcal{C}(S) = \mathcal{C}(T)$. To avoid reasoning modulo TG isomorphisms below, we define the *graph rewrite relation* $\to_{\mathcal{G}}$ induced by the GRS $\mathcal{G}$ over canonical TGs. We define $S \to_{L\to R} T$ if $S \rightsquigarrow_{\langle L'\to R',m,u\rangle} T'$ and $\mathcal{C}(T') = T$ holds for a renaming $L' \to R'$ of $L \to R$, some morphism m and node u. Renaming ensures that nodes in $L \to R$ are fresh with respect to S. Notice that independent of the particular renaming $L \to R$, the reduct T is unique. Finally, we define $S \to_{\mathcal{G}} T$ if $S \to_{L\to R} T$ holds for some rule $L \to R \in \mathcal{G}$.

Runtime Complexity

To measure the complexity of an operation $\mathtt{f} \in \mathcal{D}$ we adopt a *unitary cost model*, where each reduction step has unit cost. To this end, we look at calls to $\mathtt{f}$ when supplied with values. In short, we restrict our attention to reductions starting from *basic TGs* $\diamondsuit(\mathcal{D},\mathcal{C})$, i.e. TGs whose root is labeled by a defined symbol $\mathcal{D}$, and whose arguments are values formed from $\mathcal{C}$. Similarly, the set $\triangle(\mathcal{D},\mathcal{C}) \subseteq \diamondsuit(\mathcal{D},\mathcal{C})$ of *basic trees* is defined. We abbreviate $\diamondsuit(\mathcal{D},\mathcal{C})$ and $\triangle(\mathcal{D},\mathcal{C})$ by $\diamondsuit$ and $\triangle$, respectively.

Let S be a set of TGs, parameterised in their size. For simplicity, we assume that S denotes the limit of a family of TGs $(S_i)_{i=\mathbb{N}}$, where $S_i \subseteq S$ collects all TGs in $\bigcup_{i\in\mathbb{N}} S$ of size up to i. As above we denote the family simply by S. Recall that $\mathrm{rc}^S_{\to_{\mathcal{G}}}(n) = \max\{dh_{\to_{\mathcal{G}}}(s) \mid s \in S_n\}$, cf. page 3. The *runtime complexity (function)* of $\mathcal{G}$ with respect to *starting graphs* S is defined as $\mathrm{rc}^S_{\mathcal{G}}(n) := \mathrm{rc}^S_{\to_{\mathcal{G}}}(n)$.

Furthermore, we set $dh_{\mathcal{G}}(T) := dh_{\to_{\mathcal{G}}}(T)$. Of particular interest will be the runtime complexity $\mathrm{rc}^{\diamondsuit}_{\mathcal{G}}(n)$ of $\mathcal{G}$ on basic TGs, and the runtime complexity $\mathrm{rc}^{\triangle}_{\mathcal{G}}(n)$ of $\mathcal{G}$ on basic trees.

Relative Term Graph Rewriting

Rather than focusing solely on a GRS $\mathcal{G}$, we also consider the graph rewrite relation of a GRS $\mathcal{G}$ relative to a GRS $\mathcal{H}$, in notation $\mathcal{G}/\mathcal{H}$. This way, we can seamlessly adopt the combination framework for complexity analysis underlying our tool `TcT` [4] and the certifier `CeTA` [6]. The relation $\to_{\mathcal{G}}/\to_{\mathcal{H}}$ is abbreviated by $\to_{\mathcal{G}/\mathcal{H}}$. Note that $\to_{\mathcal{G}/\mathcal{H}}$ specialises to $\to_{\mathcal{G}}$ for the case $\mathcal{H} = \varnothing$. Similar to above, we set $dh_{\mathcal{G}/\mathcal{H}}(T) := dh_{\to_{\mathcal{G}/\mathcal{H}}}(T)$ and $\mathrm{rc}^S_{\mathcal{G}/\mathcal{H}}(n) := \mathrm{rc}^S_{\to_{\mathcal{G}/\mathcal{H}}}(n)$ for a set of TGs S. Note that the derivation height of a TG T with respect to $\to_{\mathcal{G}/\mathcal{H}}$, if defined, corresponds to the number of applications of rules from $\mathcal{G}$ in a $\mathcal{G} \cup \mathcal{H}$-derivation. Thus, intuitively, $\mathrm{rc}^S_{\mathcal{G}/\mathcal{H}}(n)$ measures the complexity of $\mathcal{G} \cup \mathcal{H}$, where applications of rules from $\mathcal{H}$ are free.

The following is a straight forward adaption of H. Zankl and M. Korp [30, Theorem 3.6].

▶ **Proposition 2.** *For three GRSs* $\mathcal{G}_1, \mathcal{G}_2$ *and* $\mathcal{H}$ *and any set of TGs* S*, we have*

$$\mathrm{rc}^S_{\mathcal{G}_1\cup\mathcal{G}_2/\mathcal{H}}(n) \leqslant_{\mathsf{k}} \mathrm{rc}^S_{\mathcal{G}_1/\mathcal{G}_2\cup\mathcal{H}}(n) + \mathrm{rc}^S_{\mathcal{G}_2/\mathcal{G}_1\cup\mathcal{H}}(n)\ , \quad \text{for all } n \in \mathbb{N}.$$

Proof of Proposition 2. Fix a TG T such that both $dh_{\mathcal{G}_1/\mathcal{G}_2\cup\mathcal{H}}(T)$ and $dh_{\mathcal{G}_2/\mathcal{G}_1\cup\mathcal{H}}(T)$ are defined, i.e., T neither admits a $\mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{H}$ derivation with infinitely many applications of

rules from $\mathcal{G}_1$ or from $\mathcal{G}_2$. Clearly, in any such reduction of T, we can estimate the number $l \in \mathbb{N}$ of applications of rule from $\mathcal{G}_1$ by $dh_{\mathcal{G}_1/\mathcal{G}_2\cup\mathcal{H}}(T)$. Likewise, the number $k \in \mathbb{N}$ of applications of rule from $\mathcal{G}_2$ is bounded by $dh_{\mathcal{G}_2/\mathcal{G}_1\cup\mathcal{H}}(T)$. In total thus, the considered reduction admits $k + l \leqslant dh_{\mathcal{G}_1/\mathcal{G}_2\cup\mathcal{H}}(T) + dh_{\mathcal{G}_2/\mathcal{G}_1\cup\mathcal{H}}(T)$ applications of rules from $\mathcal{G}_1 \cup \mathcal{G}_2$. As the considered reduction was arbitrary, this implies

$$dh_{\mathcal{G}_1\cup\mathcal{G}_2/\mathcal{H}}(T) \leqslant_{\mathsf{k}} dh_{\mathcal{G}_1/\mathcal{G}_2\cup\mathcal{H}}(T) + dh_{\mathcal{G}_2/\mathcal{G}_1\cup\mathcal{H}}(T) \,,$$

which then easily generalises to the runtime complexity function. ◀

4 An Interpretation Method for Graph Rewriting

In this section we introduce an *interpretation method* for graph rewrite systems. We start with a trivial extension of *quasi-interpretations* from terms to term graphs.

▶ **Definition 3.** An *$\mathcal{F}$-algebra* $\mathcal{A}$ for a signature $\mathcal{F}$ consists of a set A, the *carrier*, together with operations $\mathtt{f}_\mathcal{A} : A^k \to A$ for every k-ary function symbol $\mathtt{f} \in \mathcal{F}$. For an $\mathcal{F}$-algebra $\mathcal{A}$, *assignment* $\alpha : V_T \to A$ and TG T, the *interpretation* $[\![u]\!]_T^{\mathcal{A},\alpha}$ *of a node* u in T is defined as follows.

$$[\![u]\!]_T^{\mathcal{A},\alpha} := \begin{cases} \mathtt{f}_\mathcal{A}([\![u_1]\!]_T^{\mathcal{A},\alpha}, \dots, [\![u_k]\!]_T^{\mathcal{A},\alpha}) & \text{if } T(u) = \mathtt{f}(u_1, \dots, u_k), \\ \alpha(u) & \text{otherwise.} \end{cases}$$

Throughout the following, we fix the algebra $\mathcal{A}$ and write $[\![u]\!]_T^\alpha$ instead of $[\![u]\!]_T^{\mathcal{A},\alpha}$. Similar, we may drop the reference to α when T is ground. Note that the interpretation $[\![u]\!]_T$ corresponds to the interpreting of the term obtained unfolding the subgraph $T{\restriction}u$. As such, it cannot observe sharing. This, in turn, allows us to prove the following, even in the case where matching is non-injective.

▶ **Lemma 4.** *Suppose $S \geqslant_m T$ holds for two TGs S and T, where T is ground. For all $u \in V_S$, define $\alpha(u) := [\![m(u)]\!]_T$. Then $[\![u]\!]_S^\alpha = [\![m(u)]\!]_T$ holds for all nodes u of S.*

Proof. The proof is by induction on the term graph structure of S. In the base case, where we consider a variable node u of S, we have $[\![u]\!]_S^\alpha = \alpha(u) = [\![m(u)]\!]_T$ as desired. In the inductive step, we consider a node u in S with $S(u) = \mathtt{f}(u_1, \dots, u_k)$. As by assumption the function m is morphic in u, we see that $T(m(u)) = \mathtt{f}(m(u_1), \dots, m(u_k))$. Thus by definition and induction hypothesis,

$$[\![u]\!]_S^\alpha = \mathtt{f}_\mathcal{A}([\![u_1]\!]_S^\alpha, \dots, [\![u_k]\!]_S^\alpha) = \mathtt{f}_\mathcal{A}([\![m(u_1)]\!]_T, \dots, [\![m(u_k)]\!]_T) = [\![m(u)]\!]_T \,. \qquad ◀$$

Let $>_A$ denote a *proper*, i.e. transitive and irreflexive, order on the carrier A of an $\mathcal{F}$-algebra $\mathcal{A}$, and denote by $\geqslant_A$ its reflexive closure. Then $(\mathcal{A}, >_A)$ is a *weakly monotone $\mathcal{F}$-algebra* (*WMA* for short) if all interpretations $\mathtt{f}_\mathcal{A} : A^k \to A$ are monotone with respect to $\geqslant_A$. Here $\mathtt{f}_\mathcal{A}$ is *monotone with respect to an order* $\succ$ on A if

$$a_i \succ b \implies \mathtt{f}_\mathcal{A}(a_1, \dots, a_i, \dots, a_k) \succ \mathtt{f}_\mathcal{A}(a_1, \dots, b, \dots, a_k) \,.$$

▶ **Definition 5.** A WMA $(\mathcal{A}, >_A)$ is called a *quasi-model* for a GRS $\mathcal{G}$ if $[\![l]\!]_G^{\mathcal{A},\alpha} \geqslant_A [\![r]\!]_G^{\mathcal{A},\alpha}$ holds for all rules $(G, l, r) \in \mathcal{G}$ and all assignments $\alpha : V_G \to A$.

By weak monotonicity, the quasi-model condition on $\mathcal{G}$ extends to $\to_\mathcal{G}$. More precise, the following lemma holds. Here, for a step $S \rightsquigarrow_{\langle L\to R,m,u\rangle} T$, where by definition $T = S\langle m(R)\rangle_u$, we say that a node $w \in N_T$ *originates from a node* v in S if either $w = v \in N_S$, or $w = \mathsf{rt}(m(R))$ and $v = u$. In particular, the root of T always originates from the root of S.

▶ **Lemma 6.** *Let $(\mathcal{A}, >_A)$ be a quasi-model for a GRS $\mathcal{G}$, and consider a step $S \to_{\mathcal{G}} T$. Then $[\![v]\!]_S \geqslant_A [\![w]\!]_T$ holds for every node $w \in N_T$ that originates from a node $v \in N_S$.*

Proof. Suppose $S \rightsquigarrow_{\langle L \to R, m, u\rangle} T$ for a renaming $L \to R$ of a rule in $\mathcal{G}$, and fix a node $w \in N_T$ that originates from $v \in N_S$. The only non-trivial case is when the node w lies along the path from the root of T to the root of the plugged graph $m(R)$, as otherwise $T|w = S|v$ and thus trivially $[\![v]\!]_S = [\![w]\!]_T$.

Hence fix a node w along this path. The proof is by induction on the distance of w to the $\mathsf{rt}(m(R))$. In the base case, $w = \mathsf{rt}(m(R))$. We distinguish two cases. In the first case, w is a node of S, and thus w originates from itself. Thus $L \to R$ is a collapsing rule which implies $S|w = T|w$, and hence the claim follows in this case. Otherwise, $w = \mathsf{rt}(m(R)) = \mathsf{rt}(R)$ is a fresh node and w originates from the redex node u. Recall that by construction, the right-hand side R is embedded via $\underline{m}$ in T at node $\mathsf{rt}(R)$, i.e. $R \geqslant_{\underline{m}} T|\mathsf{rt}(R)$ holds. Define the assignment α by $\alpha(v') := [\![m(v')]\!]_S = [\![m(v')]\!]_T$ for all variable nodes v' of L. As we also have $L \geqslant_m S|u$, two applications of Lemma 4 and the quasi-model condition yields:

$$[\![u]\!]_S = [\![m(\mathsf{rt}(L))]\!]_S = [\![\mathsf{rt}(L)]\!]^\alpha_L \geqslant_A [\![\mathsf{rt}(R)]\!]^\alpha_R = [\![\underline{m}(\mathsf{rt}(R))]\!]_T = [\![\mathsf{rt}(R)]\!]_T \,.$$

This concludes the base case. The inductive step then follows directly from the induction hypothesis and weak monotonicity of the quasi-model. ◀

Incorporating Sharing

Our approach to sharing is simple but effective. Conceptually, the semantics imposed by a quasi-model $\mathcal{A}$ are used to associate a notion of *size* to term graphs. An alternative interpretation $\mathcal{B}$ on operation symbols $\mathtt{f} \in \mathcal{D}$ can then be used to measure the complexity of calls to $\mathtt{f}$, where the *size* of the arguments is given by $\mathcal{A}$. A term graph T is then interpreted by summing up the interpretation of all calls to defined symbols in T. Conditions put on rewrite rules then ensure that T interpreted gives a bound on the length of reductions of T. However, as soon as we move to the dependency pair setting, the separation into two algebras $\mathcal{A}$ and $\mathcal{B}$ becomes inessential. In the following, we therefore restrict ourselves to a single algebra that is used to measure sizes as well as the complexity of function calls. This intuition is formalised as follows.

Let $\mathcal{A}$ be an $\mathcal{F}$-algebra, equipped with a binary operation $\oplus$ and constant 0_A such that $(A, \oplus, 0_A)$ forms a commutative monoid, that is, $\oplus : A \times A \to A$ is associative and commutative, with identity 0_A. Then $(\mathcal{A}, \oplus, 0_A)$ is called an *abelian* $\mathcal{F}$-algebra. Furthermore, if $(\mathcal{A}, >_A)$ is a WMA, and $\oplus$ is monotone with respect to $>_A$ (and hence also with respect to $\geq_A$), then $((\mathcal{A}, \oplus, 0_A), >_A)$ is called a *weakly-monotone abelian algebra* (*WMAA* for short). Notice that addition ($\oplus$) extends in the obvious way to summation $\sum$ over finite multisets over A, in particular, the summation over an empty set is 0_A.

▶ **Definition 7.** Let $(\mathcal{A}, \oplus, 0_A)$ be an abelian $\mathcal{F}$-algebra. For a TG T, an assignment $\alpha : V_T \to A$ we define the *$\mathcal{D}$-interpretation* $[\![T]\!]^{\mathcal{A},\alpha}_{\mathcal{D}}$ *of* T by

$$[\![T]\!]^{\mathcal{A},\alpha}_{\mathcal{D}} := \sum_{u \in N^{\mathcal{D}}_T} [\![u]\!]^{\mathcal{A},\alpha}_T \,.$$

As before, we drop the index $\mathcal{A}$ in $[\![T]\!]^{\mathcal{A},\alpha}_{\mathcal{D}}$ when clear from context, and the assignment α for ground term graphs. Note that as a particular consequence of Lemma 4, the interpretation of isomorphic term graphs coincides.

▶ **Lemma 8.** *Let S and T be two ground TGs. If $S \cong T$ then $[\![S]\!]_{\mathcal{D}} = [\![T]\!]_{\mathcal{D}}$.*

Proof. By assumption, there exists a bijective morphism $S \geqslant_m T$. Lemma 4 yields $[\![u]\!]_S = [\![m(u)]\!]_T$, for all nodes u of S. Since m is bijective and respects the labeling of nodes, we conclude

$$[\![S]\!]_{\mathcal{D}} = \sum_{u \in N_S^{\mathcal{D}}} [\![u]\!]_S = \sum_{u \in N_S^{\mathcal{D}}} [\![m(u)]\!]_T = \sum_{v \in N_T^{\mathcal{D}}} [\![v]\!]_T = [\![T]\!]_{\mathcal{D}} \,. \qquad \blacktriangleleft$$

In the following, we give a sufficient criterion for an embedding of the rewrite relation $\to_{\mathcal{G}}$ into $>_A$ via the interpretation $[\![\cdot]\!]_{\mathcal{D}}$, that is, $S \to_{\mathcal{G}} T$ implies $[\![S]\!] >_A [\![T]\!]$. A first attempt might be to require that $[\![L]\!]^{\alpha}_{\mathcal{D}} >_A [\![R]\!]^{\alpha}_{\mathcal{D}}$ holds for all rules $L \to R \in \mathcal{G}$ and assignments α. The following example clarifies that such an orientation of rewrite rules is not sufficient, even when $(\mathcal{A}, \oplus, 0_A)$ is a quasi-model for $\mathcal{G}$.

▶ **Example 9.** Consider the one-rule GRS $\mathcal{G}_1 := \{\mathtt{f}(\mathtt{g}) \to \mathbf{c}\}$, over a signature $\mathcal{F}_1$ consisting of defined symbols $\mathcal{D} = \{\mathtt{f}, \mathtt{g}\}$, the constructor $\mathbf{c}$, and an additional binary constructor $\mathbf{d}$. We consider the WMAA $((\mathcal{A}_1, +, 0), >_{\mathbb{N}})$ over $\mathbb{N}$, where the interpretation $\mathcal{A}_1$ is defined by:

$$\mathtt{g}_{\mathcal{A}_1} := 1 \qquad \mathtt{f}_{\mathcal{A}_1}(x) := 0 \qquad \mathbf{c}_{\mathcal{A}_1} := 0 \qquad \mathbf{d}_{\mathcal{A}_1}(x, y) := 0 \,.$$

Then $((\mathcal{A}_1, +, 0), >_{\mathbb{N}})$ constitutes a quasi-model for $\mathcal{G}_1$. Let G denote the graph underlying the rule $\mathtt{f}(\mathtt{g}) \to \mathbf{c}$, and let $u_{\mathtt{g}}$ be the node labeled by $\mathtt{g}$ in G. Then

$$[\![\mathtt{f}(\mathtt{g})]\!]^{\mathcal{A}_1}_{\mathcal{D}} = \mathtt{f}_{\mathcal{A}_1}([\![u_{\mathtt{g}}]\!]_G) + \mathtt{g}_{\mathcal{A}_1} = 0 + 1 = 1 >_{\mathbb{N}} 0 = [\![\mathbf{c}]\!]^{\mathcal{A}_1}_{\mathcal{D}} \,.$$

On the other hand, for a binary constructor $\mathbf{d}$, the GRS $\mathcal{G}_1$ gives rise to a rewrite step

$$S := \quad \begin{array}{ccc} & \mathbf{d} & \\ \swarrow & & \searrow \\ \mathtt{f} & & \mathtt{g}^{v} \\ \searrow & & \swarrow \\ & \mathtt{g} & \end{array} \quad \to_{\mathcal{G}_1} \quad \begin{array}{ccc} & \mathbf{d} & \\ \swarrow & & \searrow \\ \mathbf{c} & & \mathtt{g} \end{array} \quad =: T \,.$$

However this step is not embedded into $>_{\mathbb{N}}$:

$$[\![S]\!]^{\mathcal{A}_1}_{\mathcal{D}} = \mathtt{f}_{\mathcal{A}_1}([\![v]\!]_S) + \mathtt{g}_{\mathcal{A}_1} = 1 \not> 1 = \mathtt{g}_{\mathcal{A}_1} = [\![T]\!]^{\mathcal{A}_1}_{\mathcal{D}} \,.$$

The inequality $[\![L]\!]^{\alpha}_{\mathcal{D}} >_A [\![R]\!]^{\alpha}_{\mathcal{D}}$ is not suitably reflecting upon the replacements of nodes underlying a step $S \rightsquigarrow_{\langle L \to R, m, u \rangle} T$, in the presence of sharing. Although in the above example the shared node labeled by $\mathtt{g}$ of the reduct lies within the matched pattern but not in the right-hand side of the applied rule, it does not vanish in the reduct.

We overcome this issue via the notion of difference set, introduced in Section 3, that characterises the node replacements underlying a rewrite step (see Proposition 1).

▶ **Definition 10.** Let $(\mathcal{A}, \oplus, 0_A)$ be an abelian $\mathcal{F}$-algebra and fix an order $\succ$ on the carrier A. We say that a rule $L \to R = (G, l, r)$ *is oriented by* $\succ$ (with respect to the algebra $\mathcal{A}$ and defined symbols $\mathcal{D}$) if

$$[\![l]\!]^{\mathcal{A},\alpha}_G \succ \sum_{u \in \Delta^{\mathcal{D}}(L \to R)} [\![u]\!]^{\mathcal{A},\alpha}_G \quad \text{holds for all assignments } \alpha.$$

A GRS $\mathcal{G}$ is called *oriented by* $\succ$ if all rules in $\mathcal{G}$ are oriented by $\succ$.

The following constitutes the main technical lemma of this section.

▶ **Lemma 11.** *Let $((\mathcal{A}, \oplus, 0_A), >_A)$ be an abelian quasi-model for a GRS $\mathcal{G}$ and suppose the rule $L \to R \in \mathcal{G}$ is oriented by some $\succ \in \{\geqslant_A, >_A\}$. Then*

$$S \to_{L\to R} T \implies [\![S]\!]_{\mathcal{D}} \succ [\![T]\!]_{\mathcal{D}} \,.$$

Proof. Fix $\succ \in \{\geqslant_A, >_A\}$ and a rule $L \to R = (G, l, r)$ which is oriented by $\succ$:

$$[\![l]\!]_G^{\mathcal{A},\alpha} \succ \sum_{v\in\Delta^{\mathcal{D}}(L\to R)} [\![v]\!]_G^{\mathcal{A},\alpha} \,. \tag{$\star$}$$

We prove that for a pre-reduction step $S \rightsquigarrow_{\langle L\to R,m,u\rangle} T$, we have $[\![S]\!]_{\mathcal{D}} \succ [\![T]\!]_{\mathcal{D}}$. As Lemma 4 allows us to lift the inequality ($\star$) to renamings of $L \to R$, and since the interpretation of isomorphic term graphs coincides (Lemma 8), the lemma follows from this. Define P as the restriction of nodes $N_S \setminus \{u\}$ to nodes in T, and define $Q := \{\underline{m}(v) \mid v \in \Delta^{\mathcal{D}}(L \to R)\}$. Thus $N_T^{\mathcal{D}} \subseteq P \cup Q$, by Proposition 1. By Lemma 6 we have

$$[\![v]\!]_S \geqslant_A [\![v]\!]_T \qquad \text{for all } v \in P. \tag{$\dagger$}$$

Furthermore, since $R \geqslant_{\underline{m}} T|\underline{m}(\mathsf{rt}(R))$, by Lemma 4 we see that

$$[\![v]\!]_G^{\alpha} = [\![\underline{m}(v)]\!]_T \qquad \text{for all } v \in \Delta^{\mathcal{D}}(L \to R). \tag{$\ddagger$}$$

We conclude:

$$\begin{aligned}
[\![S]\!]_{\mathcal{D}} \geqslant_A \sum_{v\in P} [\![v]\!]_S \oplus [\![u]\!]_S &= \sum_{v\in P} [\![v]\!]_S \oplus [\![l]\!]_G^{\alpha} && \text{by definition and Lemma 4}\\
&\succ \sum_{v\in P} [\![v]\!]_S \oplus \sum_{v\in\Delta^{\mathcal{D}}(L\to R)} [\![v]\!]_G^{\alpha} && \text{using the assumption } (\star)\\
&\geqslant_A \sum_{v\in P} [\![v]\!]_T \oplus \sum_{v\in\Delta^{\mathcal{D}}(L\to R)} [\![\underline{m}(v)]\!]_T && \text{using Equalities } (\dagger) \text{ and } (\ddagger)\\
&= \sum_{v\in P} [\![v]\!]_T \oplus \sum_{v\in Q} [\![v]\!]_T = [\![T]\!]_{\mathcal{D}} && \underline{m} \text{ is injective on } \Delta^{\mathcal{D}}(L \to R). \;\blacktriangleleft
\end{aligned}$$

The following is then a straight forward consequence of Lemma 11.

▶ **Theorem 12.** *Let $((\mathcal{A}, \oplus, 0_A), >_A)$ be an abelian quasi-model for the GRSs $\mathcal{G}$ and $\mathcal{H}$. If $\mathcal{G}$ is oriented by $>_A$ and $\mathcal{H}$ is oriented by $\geqslant_A$, then $dh_{\mathcal{G}/\mathcal{H}}(T) \leqslant_{\mathsf{k}} dh_{>_A}([\![T]\!]_{\mathcal{D}}^{\mathcal{A}})$ holds for every term graph T.*

Proof. Fix a TG T with $dh_{>_A}([\![T]\!]_{\mathcal{D}}^{\mathcal{A}})$ defined. It suffices to show that every sequence of TGs $T = T_0, T_1, T_2, \ldots$ such that

$$T = T_0 \quad \to_{\mathcal{H}}^* \cdot \to_{\mathcal{G}} \cdot \to_{\mathcal{H}}^* \quad T_1 \quad \to_{\mathcal{H}}^* \cdot \to_{\mathcal{G}} \cdot \to_{\mathcal{H}}^* \quad T_2 \quad \to_{\mathcal{H}}^* \cdot \to_{\mathcal{G}} \cdot \to_{\mathcal{H}}^* \quad \ldots \,.$$

is bounded in length by $dh_{>_A}([\![T]\!]_{\mathcal{D}}^{\mathcal{A}})$. Using the assumptions on $\mathcal{G}$ and $\mathcal{H}$, Lemma 11 translates the above sequence to

$$[\![T]\!]_{\mathcal{D}}^{\mathcal{A}} = [\![T_0]\!]_{\mathcal{D}}^{\mathcal{A}} \quad \geqslant_A^* \cdot >_A \cdot \geqslant_A^* \quad [\![T_1]\!]_{\mathcal{D}}^{\mathcal{A}} \quad \geqslant_A^* \cdot >_A \cdot \geqslant_A^* \quad [\![T_2]\!]_{\mathcal{D}}^{\mathcal{A}} \quad \geqslant_A^* \cdot >_A \cdot \geqslant_A^* \quad \ldots \,.$$

As $\geqslant_A^* \cdot >_A \cdot \geqslant_A^* = >_A^+$, the claim is then easy to establish by definition of $dh_{>_A}$. ◀

We emphasise that in conjunction with Proposition 2, the theorem can be applied in an iterative fashion, moving successively rules from $\mathcal{G}$ two $\mathcal{H}$ until $\mathcal{G}$ is empty (see Example 15 below).

Polynomial Term Graph Interpretations

We now instantiate Theorem 12 to make it applicable in the context of *polynomial runtime complexity analysis*. With respect to term rewrite systems, various forms of interpretation have been used to determine quantitative properties, most prominently, restricted forms of *polynomial* [8] and *matrix interpretations* [22, 19]. Via Theorem 12, these techniques extend naturally to graph rewrite systems. For brevity, we focus here on polynomial interpretations into the naturals.

▶ **Lemma 13.** *Let $\mathcal{A}$ be an $\mathcal{F}$-algebra with carrier $\mathbb{N}$, such that every interpretation function $\mathtt{f}_\mathcal{A}$ is given by a polynomial of degree $k \in \mathbb{N}$. Then the following properties hold.*

1. *Suppose $\mathbf{c}_\mathcal{A}(x_1, \dots, x_k) \leqslant \sum_{1 \leqslant i \leqslant n} x_i + \delta$ holds for every $\mathbf{c} \in \mathcal{C}$ and for some $\delta \in \mathbb{N}$. Then there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ of degree k such that $dh_{>_\mathbb{N}}([\![T]\!]^\mathcal{A}_\mathcal{D}) \leqslant p(|T|)$ holds for every basic tree $T \in \triangle$.*
2. *Suppose $\mathbf{c}_\mathcal{A}(x_1, \dots, x_k) \leqslant \max_{1 \leqslant i \leqslant n} x_i + \delta$ holds for every $\mathbf{c} \in \mathcal{C}$ and for some $\delta \in \mathbb{N}$. Then there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ of degree k such that $dh_{>_\mathbb{N}}([\![T]\!]^\mathcal{A}_\mathcal{D}) \leqslant p(|T|)$ holds for every basic TG $T \in \diamondsuit$.*

Proof. Fix a TG $T \in \diamondsuit$ with root r, thus $T(r) = \mathtt{f}(u_1, \dots, u_k)$ for some $\mathtt{f} \in \mathcal{D}$ and some nodes u_i $(1 \leqslant i \leqslant k)$. Note that $dh_{>_\mathbb{N}}(n) = n$, as moreover only the root of T is labeled by a defined symbol, we conclude

$$dh_{>_\mathbb{N}}([\![T]\!]^\mathcal{A}_\mathcal{D}) = [\![T]\!]^\mathcal{A}_\mathcal{D} = [\![r]\!]_T = \mathtt{f}_\mathcal{A}([\![u_1]\!]_T, \dots, [\![u_k]\!]_T) \,.$$

Define $\gamma \in \mathbb{N}$ as the maximal constant $\delta \in \mathbb{N}$ occurring in the interpretation $\mathbf{c}_\mathcal{A}$ of a constructor. By assumption on $\mathtt{f}_\mathcal{A}$, we conclude Property 1 by observing that $[\![u_i]\!]_T \leqslant |T{\restriction}u_i| \cdot \delta$ holds for all $1 \leqslant i \leqslant k$ whenever $T \in \triangle$. This follows by a standard induction on $T{\restriction}u_i$. Note that in the inductive step we makes essential use of the tree shape of T. Concerning Property 2, the form put on interpretations of constructors allows us to dispense the assumption $T \in \triangle$. Indeed, here $[\![u_i]\!]_T$ is bounded by a linear function in the depth of the graphs $T{\restriction}u_i$ $(1 \leqslant i \leqslant k)$. ◀

If the pre-conditions of Lemma 13(1) (Lemma 13(2), respectively) are satisfied, we call $\mathcal{A}$ a *$\triangle$-restricted* (*$\diamondsuit$-restricted*) *polynomial interpretation* of degree k. The following, then, is a consequence of Theorem 12 and Lemma 13. In essence, a $\triangle$-restricted polynomial interpretation permits the interpretation of values linearly in their size, whereas $\diamondsuit$-restricted polynomial interpretations measures values in their depth.

▶ **Corollary 14.** *Let $((\mathcal{A}, +, 0), >_\mathbb{N})$ be an abelian quasi-model for GRSs $\mathcal{G}$ and $\mathcal{H}$. Suppose $\mathcal{G}$ is oriented by $>_\mathbb{N}$ and $\mathcal{H}$ is oriented by $\geqslant_\mathbb{N}$ (with respect to defined symbols $\mathcal{D}$).*

1. *If $\mathcal{A}$ is a $\triangle$-restricted polynomial interpretation of degree k, then $\mathrm{rc}^\triangle_{\mathcal{G}/\mathcal{H}}(n) \in \mathrm{O}(n^k)$.*
2. *If $\mathcal{A}$ is a $\diamondsuit$-restricted polynomial interpretation of degree k, then $\mathrm{rc}^\diamondsuit_{\mathcal{G}/\mathcal{H}}(n) \in \mathrm{O}(n^k)$.*

▶ **Example 15.** Consider the following GRS that flattens trees to lists:

$$\begin{array}{lll} 1\colon \mathtt{flatten}(\mathbf{l}) \to [\,] & 2\colon \mathtt{flatten}(\mathbf{n}(e,s,t)) \to e : (\mathtt{flatten}(s) +\!\!+ \mathtt{flatten}(t)) \\ 3\colon \quad [\,] +\!\!+ ys \to ys & 4\colon \quad (x : xs) +\!\!+ ys \to x : (xs +\!\!+ ys) \,. \end{array}$$

Collect in $\mathcal{G}_\mathtt{flatten}$ the rules defining $\mathtt{flatten}$, likewise collect in $\mathcal{G}_{+\!\!+}$ the ones defining $+\!\!+$. Define the $\triangle$-restricted polynomial interpretation $\mathcal{A}_2$ such that

$$\begin{array}{lll} \mathbf{l}_{\mathcal{A}_2} := 1 & \mathbf{n}_{\mathcal{A}_2}(e,s,t) := 1 + s + t & \mathtt{flatten}_{\mathcal{A}_2}(t) := t \\ [\,]_{\mathcal{A}_2} := 0 & x :_{\mathcal{A}_2} xs := 1 + xs & xs +\!\!+_{\mathcal{A}_2} ys := xs + ys \end{array}$$

Then it can be verified that $((\mathcal{A}_2, +, 0), >_{\mathbb{N}})$ is a quasi-model for the considered GRS, and moreover, orients the rules in $\mathcal{G}_{\texttt{flatten}}$ strictly, and rules from $\mathcal{G}_{\#}$ weakly. Note that $\mathcal{A}_2$ is a $\triangle$-restricted polynomial interpretation of degree 1. By Proposition 2 and Corollary 14, the runtime complexity of $\mathcal{G}$ on trees is bounded by $\mathrm{O}(n) + \mathrm{rc}^{\triangle}_{\mathcal{G}_{\#}/\mathcal{G}_{\texttt{flatten}}}(n)$. Now define a second quasi-model $\mathcal{A}_3$ like $\mathcal{A}_2$, but with $[\,]_{\mathcal{A}_3} := 1$ and $xs \mathbin{\#}_{\mathcal{A}_3} ys := xs$ instead. This interpretation orients rules from $\mathcal{G}_{\#}$ strictly, and rules from $\mathcal{G}_{\texttt{flatten}}$ weakly, and thus $\mathrm{rc}^{\triangle}_{\mathcal{G}_{\#}/\mathcal{G}_{\texttt{flatten}}}(n) \in \mathrm{O}(n)$ by Corollary 14. Conclusively, the overall runtime is linear on trees.

Note that neither of the two interpretations is a $\diamondsuit$-restricted polynomial interpretation, due to the interpretation of the constructor $\mathbf{n}$. Indeed, the runtime complexity of the system on general term graphs is exponential, e.g., consider the flattening of a fully collapsed graph.

5 Dependency Pairs for Complexity Analysis

In the following, we suite *dependency tuples* [23], a variant of dependency pairs admissible for the innermost runtime complexity analysis of term rewrite systems, to graph rewrite systems. In the context of term graph rewriting, we will see that soundness of the method is independent of a particular reduction strategy.

For each k-ary defined symbol $\mathtt{f} \in \mathcal{D}$, let $\mathtt{f}^\sharp$ denote a fresh function symbol also of arity k, the *dependency pair symbol* of $\mathtt{f}$. Marked defined symbols are collected in $\mathcal{D}^\sharp$. Furthermore, let $\mathcal{C}\mathsf{om}$ denote the countable infinite set of *compound symbols* c_k for all $k \in \mathbb{N}$. The arity of c_k is k. For a TG T, symbol $\mathtt{f}$ and nodes $\{u_1, \dots, u_{\mathsf{ar}(\mathtt{f})}\} \in N_T$, we write $\mathtt{f}(T{\restriction}u_1, \dots, T{\restriction}u_{\mathsf{ar}(\mathtt{f})})$ for the term graph $S{\restriction}u_{\mathtt{f}}$, where S is defined as the extension of the TG T by a fresh node $u_{\mathtt{f}} \notin N_T$ with $S(u_{\mathtt{f}}) = \mathtt{f}(u_1, \dots, u_{\mathsf{ar}(\mathtt{f})})$. For a TG T rooted in a defined symbol $\mathtt{f} \in \mathcal{D}$, i.e. $T(\mathsf{rt}(T)) = \mathtt{f}(u_1, \dots, u_k)$, the *marking* $T^\sharp$ of T is defined as $\mathtt{f}^\sharp(T{\restriction}u_1, \dots, T{\restriction}u_{\mathsf{ar}(\mathtt{f})})$. This notation is naturally extended to sets.

▶ **Definition 16.** Let $L \to R$ be a rule with $\Delta^{\mathcal{D}}(L \to R) = \{u_1, \dots, u_k\}$. Then the rule

$$\mathsf{DP}(L \to R) := L^\sharp \to \mathsf{c}_k((R{\restriction}u_1)^\sharp, \dots, (R{\restriction}u_k)^\sharp) \;,$$

is called the *dependency pair of* $L \to R$ (*DP* for short). We collect in $\mathsf{DP}(\mathcal{G})$ for each rule $L \to R$ a corresponding dependency pair $\mathsf{DP}(L \to R)$.

Kindly observe that according to the definition we only consider those subgraph $R{\restriction}u_i$ of the right-hand side R, that are outside of the interface of the rule $L \to R$. This is akin to a similar condition for dependency pairs in termination of term rewrite systems, first observed by Dershowitz [14]. Further, observe that the definition of dependency pairs is devoid of an intermediate sharing or collapsing step. I.e. a shared node in $\mathsf{DP}(L \to R)$ will have been shared in $L \to R$ already. However, utilising the notion of *tops* [25] an alternative definition could be formalised in a straightforward way. This however, would render the step-by-step simulation shown below impossible.

▶ **Example 17.** Reconsider our motivating example from the introduction, depicted in Figure 1. Represent positive integers using two unary constructors $\mathbf{0}$, $\mathbf{1}$ and a constant ϵ. Then the definition of power is expressible as the GRS $\mathcal{G}_{\texttt{power}}$ consisting of the following rules:

$$\begin{aligned}
&\texttt{power}(x, \mathbf{0}(\epsilon)) \to \mathbf{1}(\epsilon)\\
&\texttt{power}(x, \mathbf{0}(n)) \to y * y \;\; \textit{where}\; y = \texttt{power}(x, n)\\
&\texttt{power}(x, \mathbf{1}(n)) \to y * (y * x) \;\; \textit{where}\; y = \texttt{power}(x, n)\;.
\end{aligned}$$

In the last two rules, the *where*-clause indicates that the recursive call is shared, i.e. represented by the node y. For brevity, we leave multiplication abstract, however, in the following we consider the symbol $(*)$ defined. Thus $\mathsf{DP}(\mathcal{G}_{\mathtt{power}})$ consists of the following three rules:

$$\mathtt{power}^\sharp(x, \mathbf{0}(\epsilon)) \to \mathbf{c}_0$$
$$\mathtt{power}^\sharp(x, \mathbf{0}(n)) \to \mathbf{c}_2(\mathtt{power}^\sharp(x, n), y *^\sharp y) \text{ where } y = \mathtt{power}(x, n)$$
$$\mathtt{power}^\sharp(x, \mathbf{1}(n)) \to \mathbf{c}_3(\mathtt{power}^\sharp(x, n), y *^\sharp (y * x), y *^\sharp x) \text{ where } y = \mathtt{power}(x, n)\ .$$

In the following, we establish a simulation of $\mathcal{G}$ via $\mathsf{DP}(\mathcal{G})/\mathcal{G}$. In this simulation, we will consider very specific term graphs over the new signature, so called *DP graphs*: a term graph U over the signature $\mathcal{F} \cup \mathcal{D}^\sharp \cup \mathcal{C}\mathsf{om}$ is called a *DP graph* if nodes above marked nodes, i.e. nodes u with $\mathsf{lab}_U(u) = \mathtt{f}^\sharp$, are labeled by compound symbols, and all nodes below are labeled by unmarked symbols $\mathcal{F}$. Thus one can adopt the intuition that a DP graph T denotes a finite sequence of term graphs whose root is marked, where the sequence itself is constructed via compound symbols. Note that DP graphs are closed under reductions:

▶ **Lemma 18.** *Let U be a DP graph. If $U \to_{\mathsf{DP}(\mathcal{G})\cup\mathcal{G}} V$ then V is again a DP graph.*

The following notion relates term graphs T to DP graphs U. In essence, it states that every potential redex in T is represented by its marked version in U. We drive our simulation precisely via this correspondence.

▶ **Definition 19.** Let T be a ground term graph, and let V be DP graph. Then V is *good* for T, in notation $T \ggg V$, if there is an injective function $d : N_T^{\mathcal{D}} \to N_V^{\mathcal{D}^\sharp}$ such that for all $u \in N_T^{\mathcal{D}}$, $(T{\restriction}u)^\sharp \cong V{\restriction}d(u)$ holds.

Observe that $T^\sharp$ is good for T, if T is a basic term graph. Furthermore, the right-hand side of $\mathsf{DP}(L \to R)$ is good for the part of R whose nodes lie in the difference set $\Delta(L \to R)$. In the proof of the following lemma, we tacitly employ that the relation $\ggg$ is closed under isomorphisms, i.e. $\cong \cdot \ggg \cdot \cong \subseteq \ggg$.

▶ **Lemma 20.** *If $S \ggg U$ and $S \to_{L\to R} T$, then there exists a term graph V with $U \to^*_{L\to R} \cdot \to_{DP(L\to R)} V$ and $T \ggg V$.*

Proof. Let $L \to R = (G, l, r)$. Fix a pre-reduction step $S \rightsquigarrow_{\langle L\to R, m, u\rangle} T$ and suppose $S \ggg U$, as witnessed by the injective mappings $d : N_S^{\mathcal{D}} \to N_U^{\mathcal{D}^\sharp}$.

Denote by $v_1, \ldots, v_k$ all nodes labeled by a defined symbol that lie strictly above the redex node u in S, i.e. $v_i \to_S^+ u$ with $v_i \in N_S^{\mathcal{D}}$ holds for all $1 \leqslant i \leqslant k$. By the assumption $S \ggg U$, the markings of $S{\restriction}v_i$ are isomorphic to $U{\restriction}d(v_i)$, i.e. $(S{\restriction}v_i)^\sharp \cong_{m_i} U{\restriction}d(v_i)$ holds. This again implies that the assumed rewrite can be carried out in $U{\restriction}d(v_i)$. More precise, $\langle L \to R, m_i \circ m, m_i(u)\rangle$ is a redex in U, where the reduct of $U{\restriction}d(v_i)$ is isomorphic to the marking of $T{\restriction}v_i$, by construction. Kindly note that the nodes $m_i(u)$ are not necessarily pairwise distinct, however if two nodes $m_i(u)$ and $m_j(u)$ are distinct, then these two nodes are parallel. Moreover the nodes $m_i(u)$ $(1 \leqslant i \leqslant k)$ lie outside of $U{\restriction}d(v)$ for all $v \in N_S^{\mathcal{D}}$ with $v \notin \{v_1, \ldots, v_k\}$. The latter is a simple fact following from $S \ggg U$ and the relative position of v to the redex node u in S, i.e. parallel or below. In conclusion, all rewrite steps on $m_i(u)$ can be carried out independently and in sequence, resulting in a DP graph W,

$$U \rightsquigarrow_{L_1\to R_1} \cdots \rightsquigarrow_{L_n\to R_n} W\ ,$$

for suitable renamings $L_j \to R_j$ of $L \to R$. By construction, we have (i) $(S{\restriction}v)^\sharp \cong W{\restriction}d(v)$ for all nodes $v \in N_S^{\mathcal{D}}$ strictly above u, and (ii) $U{\restriction}d(v) = W{\restriction}d(v)$ for all nodes $v \in N_S$ parallel or below u (including u).

Next, assume wlog. that the nodes of $\mathsf{DP}(L \to R)$ are disjoint from those of W. Observe that by (ii), $S{\restriction}u$ and $W{\restriction}d(u)$ are isomorphic, modulo marking of $d(u)$. And thus, since L and $L^\sharp$ coincides on all but the marking of the root node, we obtain our final DP graph V, with

$$W \rightsquigarrow_{\langle \mathsf{DP}(L\to R), m^\sharp, d(u)\rangle} V \ ,$$

for a suitable matching morphism $m^\sharp$. Observe that since marked nodes in W are all in parallel, and by injectivity of d, it follows that (iii) $W{\restriction}d(v) = V{\restriction}d(v)$ holds for all $v \in N_S^{\mathcal{D}} \setminus \{u\}$.

It remains to verify that V is good for T. To this end, for each node $v \in \Delta^{\mathcal{D}}(L \to R)$, let $v_\sharp$ denote the root of the subgraph $(R{\restriction}v)^\sharp$ occurring in the right-hand side of $\mathsf{DP}(L \to R)$. Define $e : N_T^{\mathcal{D}} \to N_V^{\mathcal{D}^\sharp}$ by

$$e(v) := \begin{cases} d(v) & \text{if } v \in N_S, \\ v_\sharp & \text{if } v \in N_R^{\mathcal{D}} \setminus N_L^{\mathcal{D}}. \end{cases}$$

Observe that e is injective, moreover, it is total as a consequence of Proposition 1. We perform case analysis on $v \in N_T^{\mathcal{D}}$ and show that the marking of $T{\restriction}v$ is isomorphic to $V{\restriction}e(v)$:

- If v is strictly above the redex node u in S then the claim follows from (i) and (iii).
- If v occurs parallel or strictly below the redex node u in S, then $S{\restriction}v = T{\restriction}v$, and $U{\restriction}d(v) = V{\restriction}d(v)$ by (ii) and (iii). Then $S \ggg U$ and definition of e proves the case.
- If $v = u$, then L lies below R in $L \to R$, hence again $S{\restriction}u = T{\restriction}u$. Note that in the considered case, l occurs in the difference set $\Delta^{\mathcal{D}}(L \to R)$, and consequently $l^\sharp$ lies also below the right-hand side of $\mathsf{DP}(L \to R)$. Thus also $U{\restriction}d(u) = W{\restriction}d(u) = V{\restriction}d(u)$ where the first equality holds by (ii) and the second by construction. We conclude as above.
- Suppose $v \in N_R^{\mathcal{D}} \setminus N_L^{\mathcal{D}} \subseteq \Delta^{\mathcal{D}}(L \to R)$. We have to show that the marking of $T{\restriction}v$ is isomorphic to $V{\restriction}e(v) = V{\restriction}v_\sharp$, for $v_\sharp$ the root of the graph $(R{\restriction}v)^\sharp$ that occurs in the right-hand side of $\mathsf{DP}(L \to R)$. Compare the rewrite $S \rightsquigarrow_{\langle L\to R, m, u\rangle} T$ with $W \rightsquigarrow_{\langle \mathsf{DP}(L\to R), m^\sharp, d(u)\rangle} V$. Observe that $R{\restriction}v$ is embedded at node v in T, i.e. $R{\restriction}v \geqslant_{\underline{m}} T{\restriction}v$ where moreover, $\underline{m}$ is injective on all nodes $N_{R{\restriction}v} \setminus N_L$. In a similar fashion, $(R{\restriction}v)^\sharp$ is embedded at node $v_\sharp$ in V. Since the left-hand side of $L \to R$ is isomorphic to the left-hand side of $\mathsf{DP}(L \to R)$ modulo marking of the root, and since the redexes $S{\restriction}u$ and $W{\restriction}d(u)$ are isomorphic modulo marking of $d(u)$, it is then not difficult to conclude that $T{\restriction}v$ is isomorphic to $V{\restriction}v_\sharp$, modulo marking of $v_\sharp$.

By Proposition 1, we exhausted all cases and conclude the lemma. ◀

Note that for a rule $L \to R \in \mathcal{G}$, the sequence $U \to_{L\to R}^* \cdot \to_{\mathsf{DP}(L\to R)} V$ corresponds to a relative step $U \to_{\mathsf{DP}(\mathcal{G})/\mathcal{G}} V$. Thus, Lemma 20 is directly applicable in a relative setting.

▶ **Theorem 21.** *Let $\mathcal{G}$ and $\mathcal{H}$ be GRSs and define $\mathcal{Q} := \mathsf{DP}(\mathcal{G})/\mathsf{DP}(\mathcal{H}) \cup \mathcal{G} \cup \mathcal{H}$. Then*

$$S \ggg U \implies dh_{\mathcal{G}/\mathcal{H}}(S) \leqslant_{\mathsf{k}} dh_{\mathcal{Q}}(U) \ .$$

Proof. Consider first a relative step $S \to_{\mathcal{G}/\mathcal{H}} T$, i.e. $S \to_{\mathcal{H}}^* \cdot \to_{\mathcal{G}} \cdot \to_{\mathcal{H}}^* T$, and let U be a term graph that is good for S. As a consequence of Lemma 20, we obtain a term graph V that is good for T such that $U \to_{\mathsf{DP}(\mathcal{H})/\mathcal{H}}^* \cdot \to_{\mathsf{DP}(\mathcal{G})/\mathcal{G}} \cdot \to_{\mathsf{DP}(\mathcal{H})/\mathcal{H}}^* V$, i.e. $U \to_{\mathcal{Q}} V$, holds. From this, we conclude by following the structure of the proof of Theorem 12. ◀

Conclusively, we obtain the main result of this section.

▶ **Corollary 22.** *Let $\mathcal{G}$ and $\mathcal{H}$ be GRSs, let $\mathcal{P}$ denote the relative system $\mathcal{G}/\mathcal{H}$, and let $\mathcal{Q}$ denote the relative system $\mathsf{DP}(\mathcal{G})/\mathsf{DP}(\mathcal{H}) \cup \mathcal{G} \cup \mathcal{H}$. Then for any set $S \subseteq \Diamond$ of basic term graphs, $\mathrm{rc}_{\mathcal{P}}^S(n) \leqslant_{\mathsf{k}} \mathrm{rc}_{\mathcal{Q}}^{S^\sharp}(n)$ holds for all $n \in \mathbb{N}$.*

Suiting the Interpretation Method

Our interpretation method (Corollary 14) is readily applicable in the context of relative dependency pair systems of the form of $\mathcal{Q}$ from Corollary 22. However, the imposed constraints are unnecessarily strict, as we only need to account for steps due to dependency pairs, i.e. rewrites on marked symbols. We thus embed reductions via $[\![\cdot]\!]_{\mathcal{D}^\sharp}$ rather than $[\![\cdot]\!]_{\mathcal{D}^\sharp \cup \mathcal{D}}$. This, in turn, allows us to weaken the pre-conditions of Corollary 14. The following constitutes the central observation.

▶ **Lemma 23.** *Let $((\mathcal{A}, \oplus, 0_A), >_A)$ be a quasi-model for the GRS $\mathcal{G}$, and let U be a DP graph. Then $U \to_{\mathcal{G}} V$ implies $[\![U]\!]^{\mathcal{A}}_{\mathcal{D}^\sharp} \geqslant_A [\![V]\!]^{\mathcal{A}}_{\mathcal{D}^\sharp}$.*

Proof. Consider a step $U \to_{\mathcal{G}} V$. By the shape of U, this step has to take place strictly below marked symbols, and consequently $N := N_U^{\mathcal{D}^\sharp} = N_V^{\mathcal{D}^\sharp}$. Hence Lemma 6 yields $[\![u]\!]_U \geqslant_A [\![u]\!]_V$ for each $u \in N$, and thus $[\![U]\!]_{\mathcal{D}^\sharp} = \sum_{u \in N} [\![u]\!]_U \geqslant_A \sum_{u \in N} [\![u]\!]_V = [\![V]\!]_{\mathcal{D}^\sharp}$. ◀

Consider an abelian $\mathcal{F}$-algebra $(\mathcal{A}, \oplus, 0_A)$, and let $L \to R = (G, l, r)$ be a dependency pair with right-hand side $R = \mathsf{c}_k((R|_{u_1})^\sharp, \dots, (R|_{u_k})^\sharp)$. Then this dependency pair is oriented by an order $\succ$ on the carrier A with respect to marked defined symbols $\mathcal{D}^\sharp$ if

$$[\![l]\!]^{\mathcal{A},\alpha}_G \succ \sum_{u \in \Delta^{\mathcal{D}^\sharp}(L \to R)} [\![u]\!]^{\mathcal{A},\alpha}_G \Big(= \sum_{1 \leqslant i \leqslant k} [\![u_i]\!]^{\mathcal{A},\alpha}_G \Big) \quad \text{holds for all assignments } \alpha.$$

The following is then a simple corollary to Theorem 12, using in addition Lemma 23.

▶ **Corollary 24.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two sets of dependency pairs, and let $\mathcal{G}$ be a GRS. Let $((\mathcal{A}, \oplus, 0_A), >_A)$ be an abelian quasi-model for $\mathcal{G}$, and suppose that rules in $\mathcal{P}$ and $\mathcal{Q}$ are oriented by $>_A$ and $\geqslant_A$, respectively, with respect to the WMAA $(\mathcal{A}, \oplus, 0_A)$ and* defined symbols $\mathcal{D}^\sharp$. *Then $dh_{\mathcal{P}/\mathcal{Q} \cup \mathcal{G}}(U) \leqslant_{\mathsf{k}} dh_{>_A}([\![U]\!]_{\mathcal{D}})$ holds for every DP graph U.*

▶ **Example 25** (Continued from Example 17)**.** With the help of this final corollary, it is not difficult to bind the runtime complexity of $\mathsf{DP}(\mathcal{G}_{\mathtt{power}})/\mathcal{G}_{\mathtt{power}}$, using a $\diamondsuit$-restricted polynomial interpretation of degree one. Thus the GRS $\mathcal{G}_{\mathtt{power}}$ has linear complexity by Corollary 22, and under the assumption that multiplication has unit cost, this bound transfers to our motivating example.

6 Related Work

To the best of our knowledge this study is the first investigation towards an automated complexity analysis of *term graph rewriting*. However, in the wider scope of *graph rewriting*, complexity has been an issue.

In particular Bonfante et al. study the derivation height of specific graph rewrite systems, cf. [9]. A termination method for graph rewrite systems is established, which is based on weights. Termination induces polynomial bounds on the derivation height of the rewrite relation. As the considered graph rewrite systems always start in an initial configuration, we can roughly say that their methods establish polynomial runtime complexity of the graph computation. Due to the specific nature of the considered systems the technical results obtained in [9] cannot be compared to the results obtained in this paper. Still, the conceptional approach of proving termination of graph rewrite systems by weights and studying the induced runtime complexity is related to some degree.

In [12], H. J. Sander Bruggink, B. König and H. Zantema introduce a novel interpretation method applicable in the context of termination analysis of graph transformation systems.

Here a, possibly cyclic, graph is interpreted via its embedding into a *weighed type graph*. Sufficient orientation conditions are then put on transformation rules which ensure that the interpretation of graphs decreases during reductions. Interestingly, the method implies a linear bound on the runtime complexity of the analysed system. To overcome the limitation to such linear systems, the authors show that the method is also applicable in an iterated fashion. Then, however, sensible bounds on the runtime complexity cannot be derived. In recent work [13], H.J.S. Bruggink, B. König, D. Nolte and H. Zantema generalise the approach to type graphs over semirings. It is unclear how this generalisation relates to runtime complexity analysis, and whether it can be suited to term graphs.

Furthermore, in the literature the complexity of *interaction nets* [18] have been pondered. In contrast to term graphs considered here, interaction nets may admit cyclic structures, but on the other hand provide for more control in sharing or garbage collection, via the explicit use of duplication or erasing cells. First results on runtime complexity have been proposed by Perrinel in [24]. Furthermore, Gimenez and the second author study in [17] space and time complexities of sequential and parallel computations. The resource analysis is based on user-defined sized types in conjunction with potentials that are assigned to each cell in the net. While technically quite apart from the work presented here, there are conceptional similarities: potentials are conceivable as interpretations, and the dependency pair method implicitly combines a size analysis with a runtime analysis.

Finally, we also mention connections to [10]. Here G. Bonfante, J.-Y. Marion and J.-Y. Moyen couple a termination criterion with quasi-interpretations to derive bounds on the complexity of term rewrite systems, using memoisation to speed up computation. Partly inspired by this work, in [1] the first author together with Dal Lago introduce a machinery that incorporates sharing and memoization in order to get an even more efficient mechanism for evaluating term rewrite systems. It seems that a suitable adaptation of quasi-interpretations to term graphs, following along the lines of our adaption of polynomial interpretations, would allow the use of this machinery to strengthen the result of [10].

7 Conclusion and Future Work

In this paper we have transferred two seminal techniques in complexity analysis of term rewrite systems to term graph rewrite systems: (i) the interpretation method and (ii) the dependency pair method. Our adaptions are non-trivial, in the sense that they can observe not only term but also graph structures, i.e. take sharing into account. As our results have been obtained in the context of relative graph rewriting, we have thus established the core parts of a *complexity pair framework* for term graph rewrite systems. We expect that similar adaptions of existing processors, like for example *usable arguments* or *dependency graphs* are easily obtainable, based on the foundation provided in this paper.

An immediate concern for future work is the implementation of the proposed techniques. Using similar methods as in our existing implementation of complexity analysis of term rewrite systems [5], it is not difficult to see that all proposed methods are automatable and we do not expect any issues for the preparation of a prototype. Furthermore our motivating example highlights the interest of dedicated methods for outermost evaluation, which we want to study in the future. Also of essence is the extension of the approach to possibly cyclic graphs. More generally, one strong motivation for this work stems from our work on resource analysis of imperative programs. Existing transformations to rewrite systems, necessarily unfold the heap to a tree, as it has to be representable as a term [28]. Here we hope that the direct coupling with graph rewriting is of advantage and could provide us with a sophisticated shape analysis of the heap.

References

1 M. Avanzini and U. Dal Lago. On Sharing, Memoization, and Polynomial Time. *IC*, 2015.
2 M. Avanzini, U. Dal Lago, and G. Moser. Analysing the Complexity of Functional Programs: Higher-Order Meets First-Order. In *Proc. of 20th ICFP*, pages 152–164. ACM, 2015.
3 M. Avanzini, N. Eguchi, and G. Moser. A new Order-theoretic Characterisation of the Polytime Computable Functions. *TCS*, 585:3–24, 2015.
4 M. Avanzini and G. Moser. A Combination Framework for Complexity. *IC*, 2015.
5 M. Avanzini, G. Moser, and M. Schaper. TcT: Tyrolean Complexity Tool. In *Proc. of 22nd TACAS*, LNCS, 2016.
6 M. Avanzini, C. Sternagel, and R. Thiemann. Certification of Complexity Proofs using CeTA. In *Proc. of 26th RTA*, volume 36 of *LIPIcs*, pages 23–39, 2015.
7 E. Barendsen. Term Graph Rewriting. In *Term Rewriting Systems*, volume 55 of *CTTCS*, chapter 13, pages 712–743. Cambridge University Press, 2003.
8 G. Bonfante, A. Cichon, J.-Y. Marion, and H. Touzet. Algorithms with Polynomial Interpretation Termination Proof. *JFP*, 11(1):33–53, 2001.
9 G. Bonfante and B. Guillaume. Non-simplifying Graph Rewriting Termination. In *Proc. of 7th TERMGRAPH*, EPTCS, pages 4–16, 2013.
10 G. Bonfante, J.-Y. Marion, and J.-Y. Moyen. Quasi-interpretations: A Way to Control Resources. *TCS*, 412(25):2776–2796, 2011.
11 M. Brockschmidt, F. Emmes, S. Falke, C. Fuhs, and J. Giesl. Alternating Runtime and Size Complexity Analysis of Integer Programs. In *Proc. of 20th TACAS*, volume 8413 of *LNCS*, pages 140–155, 2014.
12 H. J. Sander Bruggink, B. König, and H. Zantema. Termination Analysis for Graph Transformation Systems. In *Proc. of 8th IFIP TC 1/WG 2.2, TCS*, volume 8705 of *LNCS*, pages 179–194, 2014.
13 H.J.S. Bruggink, B. König, D. Nolte, and H. Zantema. Proving Termination of Graph Transformation Systems Using Weighted Type Graphs over Semirings. In *Proc. of 8th ICGT*, volume 9151 of *LNCS*, pages 52–68, 2015.
14 N. Dershowitz. Termination Dependencies. In *Proc. of 6th WST*, pages 28–30. Universidad Politécnica de Valencia, 2003. Technical Report DSIC-II/15/03.
15 F. Frohn, J. Giesl, J. Hensel, C. Aschermann, and T. Ströder. Inferring Lower Bounds for Runtime Complexity. In *Proc. of 26th RTA*, LIPIcs, pages 334–349, 2015.
16 J. Giesl, T. Ströder, P. Schneider-Kamp, F. Emmes, and C. Fuhs. Symbolic Evaluation Graphs and Term Rewriting: A General Methodology for Analyzing Logic Programs. In *Proc. of 14th PPDP*, pages 1–12. ACM, 2012.
17 S. Gimenez and G. Moser. The Complexity of Interaction. In *Proc. of 43rd POPL*, pages 243–255. ACM, 2016.
18 Y. Lafont. Interaction nets. In *Proc. of 17th POPL*, pages 95–108. ACM, 1990.
19 A. Middeldorp, G. Moser, F. Neurauter, J. Waldmann, and H. Zankl. Joint Spectral Radius Theory for Automated Complexity Analysis of Rewrite Systems. In *Proc. of 4th CAI*, volume 6742 of *LNCS*, pages 1–20, 2011.
20 G. Moser. Proof Theory at Work: Complexity Analysis of Term Rewrite Systems. *CoRR*, abs/0907.5527, 2009. Habilitation Thesis.
21 G. Moser. KBOs, Ordinals, Subrecursive Hierarchies and All That. *JLC*, 2014. advance access.
22 G. Moser and A. Schnabl. Proving Quadratic Derivational Complexities using Context Dependent Interpretations. In *Proc. of 19th RTA*, volume 5117 of *LNCS*, pages 276–290, 2008.
23 L. Noschinski, F. Emmes, and J. Giesl. Analyzing Innermost Runtime Complexity of Term Rewriting by Dependency Pairs. *JAR*, 51(1):27–56, 2013.

24 M. Perrinel. On Context Semantics and Interaction Nets. In *Proc. of Joint 23rd CSL and 29th LICS*, page 73. ACM, 2014.

25 D. Plump. Simplification orders for term graph rewriting. In *Proc. 22nd MFCS*, volume 1295 of *LNCS*, pages 458–467, 1997.

26 D. Plump. Essentials of Term Graph Rewriting. *ENTCS*, 51:277–289, 2001.

27 S. L. Peyton Jones. *The Implementation of Functional Languages*. Prentice-Hall International, 1987.

28 M. Schaper. A Complexity Preserving Transformation from Jinja Bytecode to Rewrite Systems. Master's thesis, University of Innsbruck, Austria, 2014. URL: `http://cl-informatik.uibk.ac.at/users/c7031025/publications/masterthesis.pdf`.

29 J. Waldmann. Matrix Interpretations on Polyhedral Domains. In *Proc. of 26th RTA*, volume 36 of *LIPIcs*, pages 318–333, 2015.

30 H. Zankl and M. Korp. Modular Complexity Analysis for Term Rewriting. *LMCS*, 10(1:19):1–33, 2014.